



Re A
PATENT

Attorney Docket No. ASTRP.235RC2

Date: August 17, 1999

Page 1

1c511 U.S. PTO
09/375867
08/17/99



ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

ATTENTION: BOX PATENT APPLICATION

Sir:

Transmitted herewith for filing is the patent application of

Inventors: **Charles D. Lanier, Richard J. Wolf, and Leticia Villegas**

For: **INTELLIGENT HELP SYSTEM**

- (X) This application is a continuation of prior Application 08/724,947, filed October 2, 1996.
- (X) A specification in 24 pages is enclosed.
- (X) 14 sheets of drawings are enclosed.
- (X) A copy of the Reissue Declaration under 37 C.F.R. § 1.175(a) from the prior application is enclosed.
- (X) A return prepaid postcard is enclosed.
- (X) The Power of Attorney in the prior application is to the law firm of Knobbe, Martens, Olson & Bear, LLP (Customer No. 20,995).
- (X) The prior application was originally assigned to AST Research, Inc., as recorded on January 18, 1994 at Reel 6847, Frame 0109. AST Research, Inc., subsequently assigned the application to AST Innovations, Inc., as recorded on September 6, 1995 at Reel 7492, Frame 0025.
- (X) Incorporation by Reference. The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

CLAIMS AS FILED

FOR	NUMBER FILED	NUMBER EXTRA	RATE	FEE
Basic Fee			\$760	\$760
Total Claims	81 - 20 =	61 ×	\$18	\$1,098
Independent Claims	10 - 3 =	7 ×	\$78	\$546
If application contains any multiple dependent claims(s), then add		\$260		\$260
TOTAL FILING FEE				\$2,664

- (X) A check in the amount of \$2,664 to cover the filing fee is enclosed.
- (X) The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment to Account No. 11-1410. A duplicate copy of this sheet is enclosed.
- (X) Please use Customer No. 20,995 for the correspondence address.

David N. Weiss
David N. Weiss
Registration No. 41,371
Attorney of Record

APPLICATION FOR REISSUE OF U.S. PATENT NO 5,103,365

1

INTELLIGENT HELP SYSTEM

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

FIELD OF THE INVENTION

This invention relates to help systems for computers; more specifically, it relates to help systems that aid a user of a computer by providing context sensitive help.

BACKGROUND OF THE INVENTION

In order to operate a computer effectively, a user must master a number of commands and data formats. One usually accomplishes this by spending hours reading printed user documentation and/or by using trial and error techniques.

Computer-aided help system have been developed to provide on-line assistance to computer users. In response to a request by a user, those systems display help information on the display screen of the computer. Simple help systems always start with the same display, regardless of the circumstances, and the user must enter specific information to find help for his or her particular situation. More advanced help systems display context-sensitive help. Context-sensitive help systems determine what particular part of an application program the user is in. Then help information is displayed that is relevant to this user location.

While such context-sensitive help systems represent an advancement over simple help systems, they have numerous limitations. Such systems are usually tightly coupled to an application program; they must rely on the application program to keep track of and store the context. Further, since these systems are limited to displaying help information based upon program location, they will always return the same help information for a given location regardless of how the user got there. While such systems provide the convenience of on-line help, the help information they provide is nothing more than a user's manual correlated with a given program screen or function. As a result, these help systems tend to be of limited utility to the user who cannot specifically identify the problem or who has "lost his way."

SUMMARY OF THE INVENTION

The invention recognizes a need for an intelligent help system which processes information specific to the user's history, such as tasks he or she has successfully executed (and how many times) or has had previous help with, and information which defines a state of a machine and a state of a programmed application.

According to the invention, an intelligent help system for aiding the user of a computer program is provided by maintaining an historic queue and using artificial intelligence techniques to select help information based on user-directed events and the current state of the system. In particular, user-directed activities are monitored and stored in the historical queue inside a knowledge base. System states are also monitored and stored. The knowledge base is then used by an inference engine

6027700-2599750

2

to isolate the specific kind of help that a user needs. Thus, the user is given assistance upon request which is appropriate to that user's level of understanding or experience and the current activities that he or she has 5 executed.

BRIEF DESCRIPTION OF THE DRAWINGS

- FIG. 1 is a block diagram of a computer system in which the invention may be embodied.
- 10 FIG. 2 is a block diagram of a computer software system used in the preferred embodiment.
- FIG. 3A illustrates the processing of user-directed events and system states into help information.
- 15 FIG. 3B is a flow chart of the general methods of the help system.
- FIG. 4 is a flow chart of a query by the monitoring device.
- FIGS. 5A-B are a flow chart of the methods of the event interpreters.
- 20 FIG. 6 is a flow chart of the methods for processing events.
- FIG. 7 is a flow chart of the methods for identifying a task.
- 25 FIG. 8 illustrates the rule taxonomy of the invention.
- FIG. 9 illustrates frames and slots for the storage of knowledge.
- FIGS. 10A-B are a flow chart of the methods for proving rules.
- 30 FIG. 11 illustrates the operation of the display engine.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to FIG. 1, the preferred embodiment of the invention is implemented on a computer system 100 having a central processor 102, a system memory 103, a display device 105, a keyboard 106, a mouse 107, a disk memory 108, an I/O controller 101, and interconnecting means 110, such as a system bus. In the preferred embodiment, a Tandy 1000 series computer (Tandy Corporation of Ft. Worth, Tex.) is used as the system 100.

Referring to FIG. 2, a computer software system 200 is shown for programming the computer system of FIG. 1. Software system 200 is stored in system memory 103 and on disk memory 108. System 200 programs the central processor 102 to display a graphic user interface (GUI) on display monitor 105. In the preferred embodiment, help system 204 is implemented in the Tandy 50 DeskMate environment 203 which provides a software interface 205 between a user 206, a computer application 202, and an operating system 201. It will be apparent, however, that one of ordinary skill in the art, informed by this specification, could implement the invention in other operating environments.

In the preferred embodiment, an artificial intelligence (AI) paradigm is used to deal with knowledge which may be vast and uncertain, leading to multiple solutions for a given situation. An AI model has the ability to 60 learn or infer more knowledge from what it already knows. Thus, if a user requests help and the help system cannot reach a solution, the system can get further information from the user and then remember the situation; the next time that that situation occurs a solution 65 can be given without querying the user again.

Referring to FIG. 3A, help system 300 of the preferred embodiment comprises a monitoring device 320 for collecting data generated in response to user-

directed events and system states 310, a knowledge base 330 for storing data 331 along with a help information database 335 used to determine the best help to give, an inference engine 340 for interpreting data 331 and help information database 335 in knowledge base 330, and a display engine 350 for presenting appropriate help information 360 on display device 105. Data 331 comprises an historical queue 332 and a state data 333, while help information database 335 comprises a plurality of rules 334 and text 336.

FIG. 3B is a flow chart illustrating the general methods of help system 300. In step 351, user-directed events and system states are monitored. User-directed events are the activities that a user performs in an application program, for example, saving a file in a paint application or copying a block of text in a word-processing application. The system state comprises a machine state and an application-specific state. In step 352, the information collected in step 351 is stored as facts or data 331 of knowledge base 330. Specifically, sequential user-directed events are stored in historical queue 332 and knowledge about the system is stored as state data 333. In step 353 if a user requests help (e.g., pressing F1 key), then in step 355 inference engine 340 tests known data 331 against help system rules 334. However, if no help is requested in step 353, the routine loops back to the monitoring step 351.

Knowledge base 330 stores heuristics in the form of rules. Rules 334, such as those used in step 355, are premise-conclusion statements predefined by an application developer which guide inference engine 340 in selecting an output. For example, suppose a user is in a Text (word processing) listbox and no files are selected. A rule that would check this is:

35

```

if
  no files
    running.listbox TEXT
then
  help running.text

```

40

This rule attempts to "fire" by proving its premise, "no files." First, it checks the known data 331 in knowledge base 330. If this is not in data 331, it checks for other rules with this premise as their conclusion. Next, if in step 356 a match is found, the corresponding rule fires in step 357. In step 358, in response to the particular rule that fired, a conclusion is asserted and appropriate help information 360 is displayed. The format of rules in this embodiment is described hereinafter with reference to FIG. 8.

Monitoring device 320 and its functions will now be described in detail. While monitoring device 320 tracks or monitors user input, it has processing capabilities. It may assert data in knowledge base 330 after identifying a sequence of one or more events or states. It may remove or retract data concerning states which are no longer true, or reasserts a new value for old data. It keeps track of the number of times an activity has successfully been completed by a user.

Monitoring device 320 monitors different types of information, including machine states, application specific states, and historical information. The machine state includes current system level, such as within an application or accessory, running a component, or at the desktop interface. An accessory is an application that may "pop up" over another application. Examples of accessories include a pop-up calculator, calendar, or

alarm. A component is a graphic element that a user interacts with to display and accept information from a user. For example, the components in a dialogue box include radio buttons, push buttons, and edit fields. The menubar is also a component.

Monitoring device 320 also monitors application-specific states (application specifics), including information unique to an application, and component states, including information within the current machine state which is either application specific or general; since components such as radio buttons are generated in the same way, regardless of whether the component is used at the application level or general level, information about component states is also generated in a uniform manner. For example in all applications which have a menubar, choices are obtained by selecting an item off the menu.

Monitoring device 320 also tracks historical information which indicates the completion of a task for which help has been defined. This data is stored in historical queue 332. A successful completion indicates that the user no longer needs help in performing that task. More specific help information, rather than general information, can be given as the user gains more experience with a program.

The structure of historical queue 332 will now be described in detail. For each entry, an entry type is stored. For example, when a user runs a dialogue box, monitoring device 320 adds a dialogue box entry (CMP_DLG_BOX) into historical queue 332 and then stores corresponding editfield, listbox, radiobutton, iconbutton and checkbox information. Each entry into historical queue 332 can be of variable length, depending on the type of entry made. A unique "return code" is assigned to each component, thus facilitating the distinction between components. A far pointer to an entry's structure (e.g., dialogue box, component, and/or menubar structures) is stored to allow direct access to that structure. Following this, dialogue and component information is copied. The format used can be of variable length. A subentry flag is defined to indicate when there is a subentry. For example, a component may still be running while the menubar is processed, therefore, the menubar is part of a single entry. If the flag is set to a value of true, then any subentry data is stored. Another flag is defined to indicate the user keystrokes that occurred during the entry. This enables the system to determine, for example, whether the user has begun entering data into a dialogue box, or if he chose help immediately. The name of the box and menu entries, such as DLGBOX, MSGBOX, LISTBOX, or MENU, is stored to distinguish one box from another. A third flag is defined to indicate the user keystrokes that occurred before invoking help. This enables the system to determine what the user was doing in the application.

Variables which manipulate historical queue 332 are defined as follows:

ihm_HelpQ: the 500 byte queue, there is one in each task data area of the Core Service Routine (CSR).
 ihm_TOP: the physical start of the HelpQ buffer.
 ihm_ENDQ: the physical end of the HelpQ buffer.
 ihm_start_ptr: pointer to the first entry in the queue, (which is last historically).
 ihm_cur_ptr: pointer to the first byte of the last entry in the HelpQ.
 ihm_end_ptr: pointer to the last byte of data entered into the HelpQ.

ihm_save_ptr: the ptr value of the last entry made.
 saved because an invalid ptr is entered into the HelpQ
 as a NULL.
 ihm_menu_ptr: double word pointer to the current
 menubar. Updated each time mo draw is called. 5
 The data structure of the queue may be summarized
 as follows:
 dw— offset of previous entry
 dw— type
 dw— return code 10
 dw— struct ptr— offset
 dw— struct ptr— segment
 db— # subentries
 db— length of name
 db(?)— name of dig box, msg box, menu, listbox 15
 db— keyflag— any mouse or key events before entry?
 db(?)— subentry data
 db— keyflag— any mouse or key events during entry?
 dw— length of information copied or 0 20

In the preferred embodiment, the monitoring of information occurs at different places in software system 200. All state changes that result in the execution of dialogue and message boxes are monitored. In Desk-Mate environment 203, commands are entered by the 25 user through a menubar and are processed by event interpreters, which check for menubar changes. In addition, the application programs themselves may indicate state changes in their respective working areas.

DeskMate environment 203 is divided into the following hierarchy of level changes, thus simplifying the monitor's task of updating information: 30

1. Top Level to application/accessory or menubar/component;
2. Application to accessory or menubar/component 35 or return to top level;
3. Accessory to menubar/component or return to application or top level.

Specific information is required as to the following states: application is running or has quit; which application is running; accessory is running or has quit; which accessory is running; dialogue box is running or has quit; message box is running or has quit; component is running or has quit; menubar menu has been pulled down or has returned; user at desk top; if user not at top 40 level, the level attained prior to the current one. The variable "level" takes the value of DeskTop, Menubar, Dig_box, Message_box, Info_box, or Component. The monitoring can obtain the state changes by getting the address calls from DeskMate environment 203 45 which indicate a function call to an application program or resource.

FIG. 4 illustrates how monitoring device 320 queries for the address of calls it is interested in. In DeskMate environment 203, applications call core service routines 55 (CSR) to run components and dialogue, information, and message boxes. A menubar interpreter handles the processing of the menubar. Therefore, independently of an application, the calls which change the structures that the application is using can be monitored to keep 60 track of information changes.

Thus, the steps are as follows. In step 401, an entry call is made to a core service routine. In step 402, the component type is identified and then compared with a list of components that monitoring device 320 is interested in. In step 403, if the component is listed in the table, then in step 404, monitoring device 320 takes the application's parameters and its structure pointer to 65

copy data from within the application's structures. The structures are not modified. Upon exit of the service and before control is returned to the application, at step 405 monitoring device 320 again accesses data in the structures, this time for the purpose of historical information updates.

Since a distinction may be made between applications and accessories, a variable "program" is defined to indicate which particular program is running. Program 10 takes the value of the program name, which is the same identifier for help information database 335 associated with the application or accessory.

Two event interpreters pick up menubar changes. The menu selected is important for the state, while the 15 item returned is important for the history. A high-priority event interpreter picks up the menubar changes when help is requested, and a low priority event interpreter will make all changes from the menubar's return code. The high-priority interpreter examines events 20 first before any further processing by the DeskMate system. On the other hand, the low-priority interpreter examines events after they are processed by the Desk-Mate system.

FIGS. 5A-B illustrate the method of the events interpreters. In step 501, the application registers the menubar with monitoring device 320. At step 502, if the user has selected the F1 (help) key, then at step 503 the state information is updated according to what level the system is operating at (indicated by the variable 30 "level"). If level equals zero in step 504, then the menubar is the last thing changed, therefore a menubar update is needed. At step 505, the menubar interpreter makes an update. But if "level" is not zero, then step 505 is skipped. At step 506, the low priority interpreter 35 checks the event type. If, at step 507, it is menubar (level = menubar), then in step 508 the low priority interpreter does a history update, performed by taking the return code and comparing its value with the menubar menus. The string corresponding to the return code is 40 the item of interest. At step 509, the routine loops back to step 501 to await another F1 keystroke.

The rules associated with the functions of the above components are written such that if a component has a title string, this string is used to identify its help source.

45 If a component does not have a title string, then the name of the component will be used as its identifier.

Since all applications use the features of DeskMate environment 203, standard representation of data 331 and rules 334 is possible. For example, the following 50 variables can be used to indicate events and states:

```

menu.selected
menu.item.selected
dig.box.running
dig.box.focus
55 msg.box.running
cmp.running
listbox.item.selected
checkbox.item.selected

```

Application-specific information is obtained by having the application assert data into the current state data 333 in knowledge base 330. The application accomplishes this by making a call to an "Assert" function with the parameters "variable" and "value," which are pointers to strings. The variable should match a variable 60 in the rule premise, i.e., the rule must be defined beforehand. An application should assert any historical information related to its unique state configuration. Application data are removed from the current state data 333

by the application once those data are no longer true by calling a "Retract" function. Although the applications may directly assert data into knowledge base 330, they do not determine what help to give. Instead, they supply the inputs for this determination.

Historical information is obtained by using a user's I.D. to index the user's unique historical information. This historical information is updated at the end of a task completion. For example, when the user has successfully executed a copying function, monitoring device 320 recognizes this by checking that the user has selected "text" and then selected "copy" from the file menu. Having determined that the user has mastered this task, monitoring device 320 updates the user's historical information. For historical information associated with the application's specific data, the application updates the information itself by calling the function UpdateHistory with a parameter pointing to a string representing the activity just completed.

FIG. 6 summarizes the method for processing the events from the event interpreters. At step 601, events are processed by examining the mouse coordinates and event type and value. In step 602, events are identified by trying to "fire" or trigger a rule that would make a data assertion or retraction into knowledge base 330. These rules represent all conditions that must be true for data to be asserted or retracted. If a rule fires in step 603, then the event is identified, step 604. If the rule does not fire, then the event is not identified, step 605.

FIG. 7 illustrates the approach used in step 602 (FIG. 6) to identify a task (sequence of events). In step 701, a key or mouse event is examined. In step 702, if the event does not match the first premise line in any rule associated with the system level, then the event is discarded in step 704. Otherwise, in step 703, all rules that fire are placed on an Agenda, which represents the most likely task(s). At step 705, the rules are approved or disproved by examining the next key or mouse events that come in. If all the rules in the Agenda fail at step 706, then the first key analyzed is discarded and the next key is used to search for new rules, step 708; otherwise in step 707, the Agenda is cleared and the procedure loops to step 701 to examine the next key/mouse event.

Since the information monitored is dependent upon the machine state level, the data generated are divided according those levels. Monitoring device 320 attempts to assert data which apply to a given level. Each application or accessory can be considered as a distinct object (as a level) which performs certain activities, some of which are common to other objects. Therefore, it is convenient to divide the monitor's data structure into "frames." A frame contains the activities that each application is capable of, with current values and historical information. The frames have slots for storing the data used by the rules or a pointer to another frame.

In the preferred embodiment, user input and system state are analyzed by monitoring device 320 for passage as data to knowledge base 330 for storage. Commands are defined for controlling this flow of data to and from knowledge base 330. Data are stored in frames through the Assert command. Old information is deleted with the Retract command. A query is made to knowledge base 330 to attain information through a Query command. Previous data are asserted by a Reassert command.

Knowledge base 330 comprises formal (traditional data base information) and informal (heuristic) knowledge which is rule based. FIG. 8 demonstrates the rule

taxonomy. At the lower level of the rule hierarchy 800 is a single rule 804 defining a pattern-to-action goal. The pattern is known as the left-hand side of a rule, while the action is the right-hand side. The rules use the common logical operators AND, OR, and NOT, as well as Boolean operators such as IF, THEN, and ELSE. In addition, the key word TEST indicates that a comparison needs to be made. The right-hand side of a rule can have an ELSE clause, an assertion, a retraction, another rule, or a procedure call. For example, in the rule:

IF a THEN b, ELSE c

b could be an assertion which would add data b into data 331; multiple data assertions are possible. A retraction would delete data b from data 331:

IF a THEN (RETRACT b)

20 "Another rule" may be embedded as follows:

IF a then b, IF (TEST (= bc)) THEN d

An example of a procedure call would be:

25 IF a THEN (CALL HelpTutorial(a)).

Linked Rules 803 are a linking between rules which share the same conclusion. This is a design implementation intended to make the inference process easier by knowing alternative solution paths. Rule Groups 802 group rules with a common purpose. For example, rules with a conclusion indicating what specific help to give are all rules determining the goal state. A group can have an (optional) priority identifier so the most important or most specific rules can be tried out first when searching. This does not imply that a rule group will be left out of a search. Rules Classes 801 are a further conceptualization of rules. They separate individual knowledge bases, each having a unique identifier by which it is distinguished. This identifier can be used by a set of control rules which use the machine state information to indicate knowledge base access.

The premise of each rule also contains formal or informal knowledge. The informal knowledge becomes formal when a rule fires successfully. The formal knowledge is stored in and accessed from knowledge base 330 in frame structures. Frames can be made up of other frames, which can also be shared. With this data structure, it is possible to access only the frame associated with the current state information when monitoring device 320 is updating the user's historical information. The frame's slots are like any linked list, except they represent actions and attributes of an object or concepts that the frames represent. Since frames indicate the relationship between a user's activity and the associated heuristics used to interpret that activity, it is easy to access only relevant information. FIG. 9 illustrates a frame 901 with its related slots 902.

Rule Classes 801 are made up of pointers to the variables and values in the frames. The slots which contain another frame are really another group within a rule class. In a frame-based reasoning system, one selects the frame to prove by filling in slot values. The slots contain the rules. The successful completion of a frame yields a solution.

-continued

```

<id>
BEGIN RULEGROUP
<id>
IF
( <variable> IS <value> )
( <Value> )
( NOT( <variable> IS <value> ) )
( TEST(<variable> EQ <#> ) )
THEN
( <value> )
END.RULEGROUP
END.RULECLASS

```

By way of illustration and not limitation, the rules may be coded as data structures (illustrated in the C 15 language):

```

/*
/* Premise is a structure containing a single premise
/* of a rule. It is made up of the string itself, the type
/* of data it is, and the possible values it can hold.
*/
struct Premise
{
    char *pVar;
    char *pValueSet;
    char *pValue;
};

/*
/* Premises is a structure for maintaining a linked
/* list of all the premises (strings) in a single rule
*/
struct Premises
{
    struct Premise *pPremise;
    int Size;
    struct Premises *pNext;
};

/*
/* RuleClass is a structure of all RuleGroups that can
/* be associated with one another in searching
*/
struct RuleClass
{
    struct RuleGroup *pRuleGroups;
    int NumberOfGroups;
    int ID;
    struct RuleClass *pNext;
};

/*
/* RuleGroup is a structure containing all rules that
/* are considered to be associated in searching
*/
struct RuleGroup
{
    struct Rule *pRules;
    int NumSubGroups;
    int Priority;
    struct RuleGroup *pNext;
};

/*
/* Rules is a structure containing all the rules within
/* a Rule Group. They can be singular or associated
/* with one another.
*/
struct Rules
{
    struct LinkedRules *pLinkedRules;
    char *pConclusion;
    struct Rules *pNext;
};

/*
/* LinkedRules is a structure for maintaining a linked
/* list of rules which all have the same conclusion. A
/* rule is made up of premises (structures), a conclusion
/* (string) and a why statement (string) explaining
/* why a rule succeeded.
*/

```

-continued

```

struct LinkedRules
{
    struct Premises *pAllPremises;
    struct LinkedRules *pNext;
};

/* ...
 * Query is a structure for maintaining a linked list
 * of all the queries for information (strings) in the
 * knowledge base.
 */
struct Query
{
    char *pFact;
    char *pAsk;
    char *pSet;
    struct Query *pNext;
};

```

The inputs to knowledge base 330 are the data collected by monitoring device 320. The outputs of knowledge base 330 are data 331 and rules 334 which inference engine 340 selects for examination and help information text 336 which display engine 350 processes. Rules 334 are predefined by application developer. The frames indicate the relationship between the user's activity and the associated heuristics used to interpret that activity, thus making it easier to access only the information needed. The knowledge-based rules are structured so that obvious associations between rules can be incorporated within knowledge base 330 simplifying the design and the inferences from those rules.

Inference engine 340 interprets data 331 and rules 334 in knowledge base 330 to give a help solution to the user, or it intelligently asks for more information in order to obtain a solution. In the preferred embodiment, inference engine 340 operates using a backward-chaining method. This method starts with a goal state (a particular kind of help) and tries to prove it by reaching initial known data.

As FIGS. 10A-B illustrate, the steps used by inference engine 340 are as follows. In step 1001, knowledge base 330 is accessed according to the system state's level. In step 1002, depending on the system state, a group of rules (goal) is selected from knowledge base 330 and used as an hypothesis. Next in step 1003 an attempt is made to prove each rule's conclusion by proving its premise. If, in step 1004, the premise examined is the conclusion of another rule, it is pushed onto a stack (last-in first-out structure in system memory 103) and an attempt is made to prove the other rule's premise in step 1005. In step 1006, if a premise is proved, it is asserted into a working fact or data list at step 1007. Otherwise if the premise fails, a search is made in step 1008 for a rule with the same conclusion, which inference engine 340 will then try to prove. In step 1009, when all the premises in the rule have been proved, the rule has fired and the conclusion can be asserted at step 1010. In step 1011, this process is repeated until there are no more goal states to analyze or until no solution is found. In the latter case, knowledge base 330 is incomplete, therefore, inference engine 340 queries the user for more information or else explains to the user that no help is defined for the particular scenario.

The inputs to inference engine 340 are data 331 and rules 334 contained in knowledge base 330. Some of rules 334 control other knowledge bases which can be selected. Inference engine 340 first examines rules 334 to select the proper rule class 801. Since the groups 802 under the class 801 indicate if they are goal states, en-

11

gine 340 can also indicate which rules will be used as hypotheses. Final output from inference engine 340 is a help "tag" that indicates a particular help solution. In addition, the engine creates a temporary output of data asserted while proving the rules.

5

Display engine 350 is an interface between the output of inference engine 340 and help text 336 giving the user easy access to the most useful help topics. Display engine 350 provides general to in-depth help to a user. It processes the inference engine's help tag to provide 10 context-sensitive help. In addition, the help information itself has tags which are used by display engine 350 to locate further help information. This allows a user to select specific help from a subset of help information.

The help information data structure used by display 15 engine 350 is defined for each application/accessory and system resource interfaces. Help that is repeated across multiple applications is divided into groups. The help information which is given is individualized for novice, intermediate, and advanced users. A display 20 format is used which allows a user to select a single item of help from a suggested list, allow the user to continue selecting help until he wants to quit, and allow the user to search through the Help Topics for a particular topic.

25

The structure by which the help information is accessed and stored is classified according to application-/accessory or system component (requires specific name identification, not type), experience level of user, kind (autodetect or user-invoked), and topic identifier. 30 For example, a help structure (in C language) can be:

```
struct Help_Source {
    int Subject;
    int Kind;
    int HelpLevel;
    char *pTopicString};
```

35

which can be filled by:

40

Subject	equal to TEXT_SUBSTITUTION
Kind	equal to AUTO

and pTopicString pointing to "Text Substitution."

45

Referring to FIG. 11, the processing of help information in display engine 350 will now be described in detail. Help information database 335 is a database containing text 336 and rules 334 fields, i.e., help information text 336 is linked to help information rules 334. In addition, a tag 1102 is provided for representing the solution that a rule produces. Display engine 350 matches tag 1102 with a solution tag 1101 from inference engine 340. The corresponding text (from text 336) is the actual text sent as help information 360 to display 55 device 105.

The organization of rules 334 is as follows. It contains the following fields: Grp, Rule#, PorC#, Var#, Var, Val, Bind, KeyW, and Q#. Grp is a rule group. Group 0 is always the group in which rules are placed that will give a solution. If these rules cause any other rules to fire, these rules are in another group. Rule# is the number of the rule. The rules are sorted by rule number, because one may want to try to fire one rule before another. PorC# is a premise or conclusion number. The records in the table are also sorted by PorC#. The conclusion is number 0 because it is the first thing pulled out of the table when inference engine 340 starts to fire

a rule. (This is because it is doing backward chaining—start with the conclusion, and then prove the premises). Var# is an identifier for the engine to do faster searching. Each new variable added to the table has a unique identifier. The variables are DeskMate components, like a dialog box. Var, the variable field, is a string representing a variable for which a value is expected such as "running.cmp" (DISABLED FIELD). Value is the string field which contains a value. If the value is the current one for the variable, then the premise line succeeds. Bind is a binding variable. If a rule has a binding, the variable will bind to the currently known value of the variable. This eliminates repetition of rules that do the same exact thing. Only premises can bind.

KeyW is a number indicating the negation (NOT) of a premise, or a TEST or CALL. TEST will do number comparisons, and assumes the value string field is a numerical value. CALL is used to execute a pre-defined function. The parameters of the function are placed in the value field. Q# is the number in queue 332 for which a premise line test applies. If the queue number equals 0, it is assumed the premise does not use predefined data.

While the invention is described in some detail with specific reference to a single preferred embodiment and certain alternatives, there is no intent to limit the invention to that particular embodiment or those specific alternatives. For example, one skilled in the art could implement such a help system in another interface environment or without any interface environment. Backward-chaining is but one of many possible AI techniques used to process data and rules, other possible techniques include forward-chaining and rule-value methods. Input device is not limited to a keyboard and a pointing device but contemplates any means by which data enters a computer, such as by voice recognition. Help information is not limited to a specific medium but instead includes any conveyance of help information, such as graphical representations. The true scope of the invention is defined not by the foregoing description but by the following claims.

What is claimed is:

1. In a computer system, a method for aiding a user of a computer program, said method operating independent of said computer program, comprising the steps of:
 - 43 storing a help information database;
 - monitoring a series of user-directed events from an input device;
 - generating data indicating said series of user-directed events;
 - 50 storing said generated data in a knowledge base;
 - storing a plurality of rules for analyzing said generated data to determine appropriate help information;
 - detecting a request for help information from the user;
 - 55 testing said rules against said generated data using an inference engine, whereby rules which are satisfied by said data are proved rules;
 - selecting in response to the proved rules appropriate help information from said help information database and
 - displaying said selected help information to the user.
2. The method of claim 1, wherein said monitoring step further comprises monitoring a system state.
- 65 3. The method of claim 2, wherein said monitoring a system state step further comprises monitoring a machine state, an application state, an accessory state, and a component state.

help

4. The method of claim 1, wherein said monitoring step further comprises the steps of:
registering an application's menubar;
checking if the user has requested help;
updating a state information; and
updating a menubar. 5
5. The method of claim 1, wherein said testing step comprises the steps of:
(a) selecting from said plurality of rules a first group of rules corresponding to a first plurality of user-directed events; 10
(b) attempting to prove each rule in said first group of rules;
(c) if a rule is proved, storing said rule as a proved rule in a plurality of proved rules; and
(d) repeating steps (a)-(c) for a subsequent group of rules until a rule is proved. 15
6. The method of claim 5, wherein step (b) comprises attempting to match a premise with each of said first group of rules with said generated data. 20
7. The method of claim 5, wherein step (c) comprises: if a rule is proved, storing said rule as a proved rule in a plurality of linked proved rules.
8. The method of claim 1, wherein said generating data step comprises generating an historical queue of 25 said user-directed events.
9. The method of claim 1, wherein said rule storing step comprises storing premise-conclusion statements from said help information database.
10. The method of claim 1, wherein said displaying step comprises displaying textual help information to the user. 30
11. The method of claim 1, wherein said displaying step comprises displaying graphical help information to the user. 35
12. The method of claim 1, wherein said testing step comprises testing said rules against said generated data using a backward-chaining inference engine.
13. The method of claim 1, wherein said testing step comprises testing rules against said generated data using 40 a forward-chaining inference engine.
14. In a computer system, a method for aiding a user of a computer program, said method operating independent of said computer program, comprising the steps of:
storing a help information database; 45
storing a knowledge base for maintaining data;
identifying a series of user-directed events;
comparing said identified series with data stored in the knowledge base;
if said identified series is unknown to said knowledge base, asserting in said knowledge base data for indicating said unknown identified series;
if said identified series contradicts said knowledge base, retracting in said knowledge base data which contradicts said identified series; 55
if said identified series is already known to said knowledge base, reasserting in said knowledge base data for indicating said already known identified series;
storing a plurality of rules for analyzing said knowledge base to determine appropriate help information; 60
detecting a request for help information from the user;
testing said rules against said knowledge from the user; 65
testing said rules against said knowledge base using an inference engine, whereby rules which are satisfied

by data stored in the knowledge base are proved rules;

selecting in response to said testing step appropriate help information from said help information database; and

displaying said selected help information to the user.

15. A help information system for aiding a user comprising:

a computer having a processor and a memory;

10 a display device coupled to said computer;

an input device coupled to said computer;

monitoring means coupled to the input device for monitoring a sequence of user-directed events and for generating data indicating said events;

15 a knowledge base coupled to said monitoring means and stored in said memory, said knowledge base comprising said generated data, a plurality of rules for analyzing said generated data to determine appropriate help information, and a help information database for storing said appropriate help information;

inference engine means, coupled to said knowledge base, for applying said rules to said data to generate an inference engine output and

25 display engine means, coupled to said inference engine and coupled to said help information database, for interpreting said inference engine output to select appropriate help information for display by said display device to the user.

30 16. The system of claim 15, wherein said monitoring means further comprises means for monitoring a system state.

35 17. The system of claim 16, wherein said means for monitoring a system state comprises means for monitoring a machine state, an application state, an accessory state, and a component state.

40 18. The system of claim 15, wherein said monitoring means comprises means, stored in memory and operably coupled to the input device, for interpreting a series of user-directed events and performing a history update based on the series.

45 19. The system of claim 15, wherein said knowledge base further comprises an historical queue stored in memory and operably coupled to said generated data.

20. The system of claim 15, wherein said inference engine means comprises backward-chaining inference engine means.

21. The system of claim 15, wherein said inference engine means comprises forward-chaining inference engine means.

22. The system of claim 15, wherein said help information comprises textual help information.

23. The system of claim 15, wherein said help information comprises graphical help information.

24. A help information system for aiding a user comprising:

a computer having a processor and a memory;

an input device coupled to said computer;

60 a knowledge base, coupled to said memory, for maintaining data;

a plurality of rules, coupled to said memory, for analyzing said knowledge base;

means, coupled to said memory, for identifying a series of user-directed events from said input device;

means, coupled to said memory, for updating said knowledge base with said identified series;

15

means, coupled to said memory, for detecting a request for help information from the user;
a help information database, coupled to said memory,
for selecting appropriate help information;
an inference engine, coupled to said memory, for testing said rules against said knowledge base to generate a help solution tag;
a display engine, coupled to said memory, for selecting help information from said help information database using said help solution tag; and
a display for displaying said selected help information to the user.

25. The system of claim 24, wherein said means for updating said knowledge base comprises programming

20

25

30

35

40

45

50

55

60

65

16

means for instructing said processor to perform the steps of:

- comparing said identified series with data stored in the knowledge base;
- 5 if said identified series is unknown to said knowledge base, asserting data in said knowledge base data for indicating said unknown identified series;
- if said identified series contradicts said knowledge base, retracting in said knowledge base data which contradicts said identified series; and
- 10 if said identified series is already known to said knowledge base, reasserting in said knowledge base data for indicating said already known identified series.

15 [* * * *]

20

25

30

35

40

45

50

55

60

65

1 26. In a computer system, a method for selecting help
2 messages for aiding a user of a computer program comprising the
3 steps of:

4 storing a help information database;
5 monitoring a series of user-directed events from an input
6 device;

7 generating data indicating said series of user-directed
8 events;

9 storing said generated data in a knowledge base; and
10 using the data indicating said series of user-directed
11 events stored in the knowledge base to select help information
12 from said help information database.

1 27. The method of claim 26, wherein said monitoring step
2 further comprises monitoring a system state.

1 28. The method of claim 27, wherein said monitoring a
2 system state step further comprises monitoring a machine state,
3 an application state, an accessory state, and a component state.

1 29. The method of claim 26, wherein said monitoring step
2 further comprises the steps of:

3 registering an application's menubar;
4 updating state information; and
5 updating a menubar.

1 30. The method of claim 26, wherein said generating data
2 step comprises generating an historical queue of said user-
3 directed events.

1 31. The method of claim 26, wherein said using step uses
2 data indicating a series of user-directed events comprising at
3 least two user-directed events.

1 32. The method of claim 26, wherein said using step uses
2 data indicating a series of user-directed events that may
3 comprise four or more user-directed events.

1 33. The method of claim 26, 31, or 32, wherein said series
2 of user-directed events are events that are not necessarily
3 related as being part of a particular command hierarchy.

1 34. The method of claim 26, further comprising the steps
2 of:

3 storing a plurality of rules for analyzing said generated
4 data to determine appropriate help information; and wherein said
5 using step further comprises using an inference engine to test
6 said rules against the data stored in the knowledge base to
7 select appropriate help information.

1 35. A help information system for aiding a user of a
2 computer program comprising:

3 a computer having a processor and a memory;
4 an output device coupled to said computer;
5 an input device coupled to said computer;
6 monitoring means coupled to the input device for monitoring
7 a sequence of user-directed events and for generating data
8 indicating said events;

9 a knowledge base coupled to said monitoring means and stored
10 in said memory, said knowledge base comprising said generated
11 data, a plurality of rules for analyzing said generated data to
12 determine appropriate help information, and a help information
13 database for storing said appropriate help information; and
14 inference engine means, coupled to said knowledge base, for
15 applying said rules to said data to select appropriate help
16 information for output by said output device to the user.

1 36. A help information system for aiding a user of a
2 computer program comprising:

3 a computer having a processor and a memory;
4 a display device coupled to said computer;
5 an input device coupled to said computer;
6 monitoring means coupled to the input device for monitoring
7 a sequence of user-directed events and for generating data
8 indicating said events;

9 a knowledge base coupled to said monitoring means and stored
10 in said memory, said knowledge base comprising said generated
11 data, a plurality of rules for analyzing said generated data to
12 determine appropriate help information, and a help information
13 database for storing said appropriate help information;

14 inference engine means, coupled to said knowledge base, for
15 applying said rules to said data to generate inference engine
16 outputs;

17 selecting means coupled to said help information database,
18 for selecting appropriate help information in response to said
19 inference engine outputs; and

20 display engine means, coupled to said selecting means for
21 presenting said appropriate help information for display by said
22 display device to the user.

1 37. The system of claim 35 or 36, wherein said monitoring
2 means further comprises means for monitoring a system state.

1 38. The system of claim 33, wherein said means for
2 monitoring a system state comprises means for monitoring a
3 machine state, an application state, an accessory state, and a
4 component state.

1 39. The system of claim 35 or 36, wherein said monitoring
2 means comprises means, stored in memory and operably coupled to
3 the input device, for interpreting a series of user-directed
4 events and performing a history update based on the series.

1 40. The system of claim 35 or 36, wherein said knowledge
2 base further comprises an historical queue stored in memory and
3 operably coupled to said generated data.

1 41. The system of claim 35 or 36, wherein said inference
2 engine means comprises backward-chaining inference engine means.

1 42. The system of claim 35 or 36, wherein said inference
2 engine means comprises forward-chaining inference engine means.

1 43. The system of claim 35 or 36, wherein said help
2 information comprises textual help information.

1 44. The system of claim 35 or 36, wherein said help
2 information comprises graphical help information.

1 45. In a computer system, a method for aiding a user of a
2 computer program comprising the steps of:

3 storing a help information database;

4 monitoring a series of user-directed events from an input
5 device;

6 generating data indicating said series of user-directed
7 events;

8 storing said generated data in a knowledge base;

9 storing a plurality of rules for analyzing said generated
10 data to determine appropriate help information;

11 testing said rules against said generated data using an
12 inference engine, whereby rules which are satisfied by said data
13 are proved rules;

14 selecting in response to the proved rules appropriate help
15 information from said help information database; and

16 displaying said selected help information to the user.

1 46. In a computer system, a method for aiding a user of a
2 computer program comprising the steps of:

3 storing a help information database;
4 monitoring a series of user-directed events from an input
5 device;
6 generating data indicating said series of user-directed
7 events;
8 storing said generated data in a knowledge base;
9 storing a plurality of rules for analyzing said generated
10 data to determine appropriate help information;
11 detecting a request for help information from the user;
12 testing said rules against said generated data using an
13 inference engine, whereby rules which are satisfied by said data
14 are proved rules;
15 Selecting in response to the proved rules appropriate help
16 information from said help information database; and
17 displaying said selected help information to the user.

1 47. The method of claim 45 or 46, wherein said monitoring
2 step further comprises monitoring a system state.

1 48. The method of claim 43, wherein said monitoring a
2 system state step further comprises monitoring a machine state,
3 an application state, an accessory state, and a component state.

1 49. The method of claim 45 or 46, wherein said monitoring
2 step further comprises the steps of:
3 registering an application's menubar;
4 updating a state information; and
5 updating a menubar.

1 50. The method of claim 45 or 46, wherein said testing step
2 comprises the steps of:
3 (a) selecting from said plurality of rules a first group of
4 rules corresponding to a first plurality of user-directed events;
5 (b) attempting to prove each rule in said first group of
6 rules;

- 7 (c) if a rule is proved, storing said rule as a proved rule
8 in a plurality of proved rules; and
9 (d) repeating steps (a)-(c) for a subsequent group of rules
10 until a rule is proved.

1 51. The method of claim 50, wherein step (b) comprises
2 attempting to match a premise with each of said first group of
3 rules with said generated data.

1 52. The method of claim 50, wherein step (c) comprises:
2 if a rule is proved, storing said rule as a proved rule in a
3 plurality of linked proved rules.

1 53. The method of claim 45 or 46, wherein said generating
2 data step comprises generating an historical queue of said user-
3 directed events.

1 54. The method of claim 45 or 46, wherein said rule storing
2 step comprises storing premise-conclusion statements from said
3 help information database.

1 55. The method of claim 45 or 46, wherein said displaying
2 step comprises displaying textual help information to the user.

1 56. The method of claim 45 or 46, wherein said displaying
2 step comprises displaying graphical help information to the user.

1 57. The method of claim 45 or 46, wherein said testing step
2 comprises testing said rules against said generated data using a
3 backward-chaining inference engine.

1 58. The method of claim 45 or 46, wherein said testing step
2 comprises testing rules against said generated data using a
3 forward-chaining inference engine.

1 59. In a computer system, a method for aiding a user of a
2 computer program comprising the steps of:
3 storing a help information database;
4 storing a knowledge base for maintaining data;
5 identifying a series of user-directed events;
6 comparing said identified series with data stored in the
7 knowledge base;
8 if said identified series is unknown to said knowledge base,
9 asserting in said knowledge base data for indicating said unknown
10 identified series;
11 if said identified series contradicts said knowledge base,
12 retracting in said knowledge base data which contradicts said
13 identified series;
14 if said identified series is already known to said knowledge
15 base, reasserting in said knowledge base data for indicating said
16 already known identified series;
17 storing a plurality of rules for analyzing said knowledge
18 base to determine appropriate help information;
19 testing said rules against said knowledge from the user;
20 testing said rules against said knowledge base using an
21 inference engine, whereby rules which are satisfied by data
22 stored in the knowledge base are proved rules;
23 selecting in response to said testing step appropriate help
24 information from said help information database;
25 and displaying said selected help information to the user.

ABSTRACT

An intelligent help system which processes information specific to a user and a system state is described. The system incorporates a monitoring device to determine which events to store as data in an historical queue. These data, as well as non-historical data (e.g., system state), are stored in a knowledge base. An inference engine tests rules against the knowledge base data, thereby providing a help tag. A display engine links the help tag with an appropriate solution tag to provide help text for display.

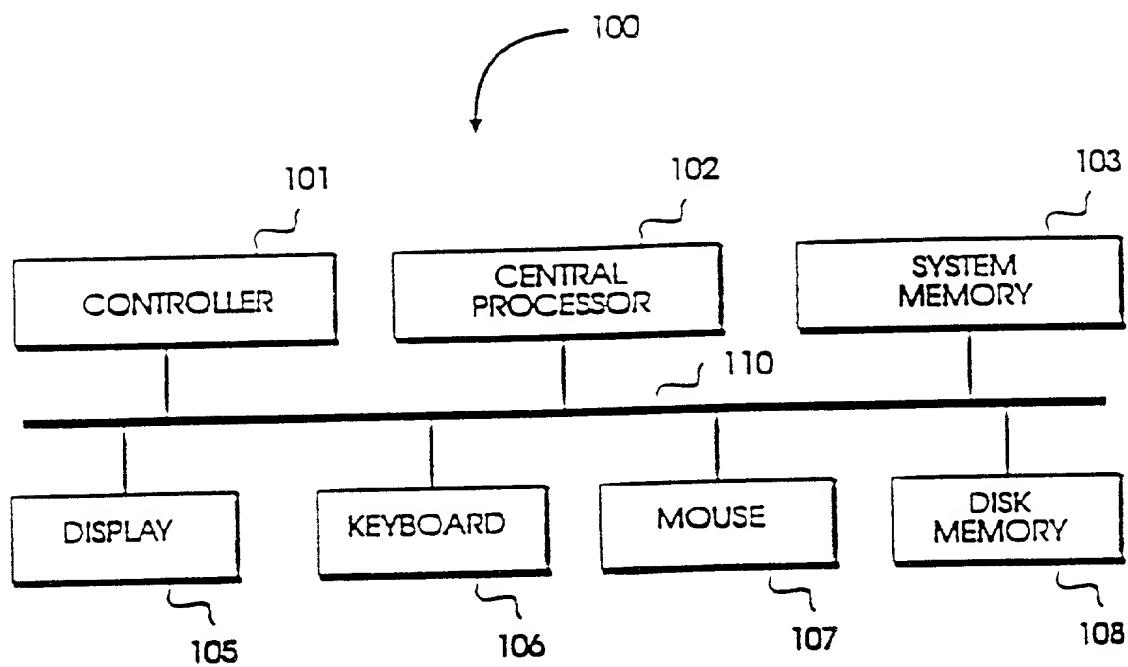


FIG. 1

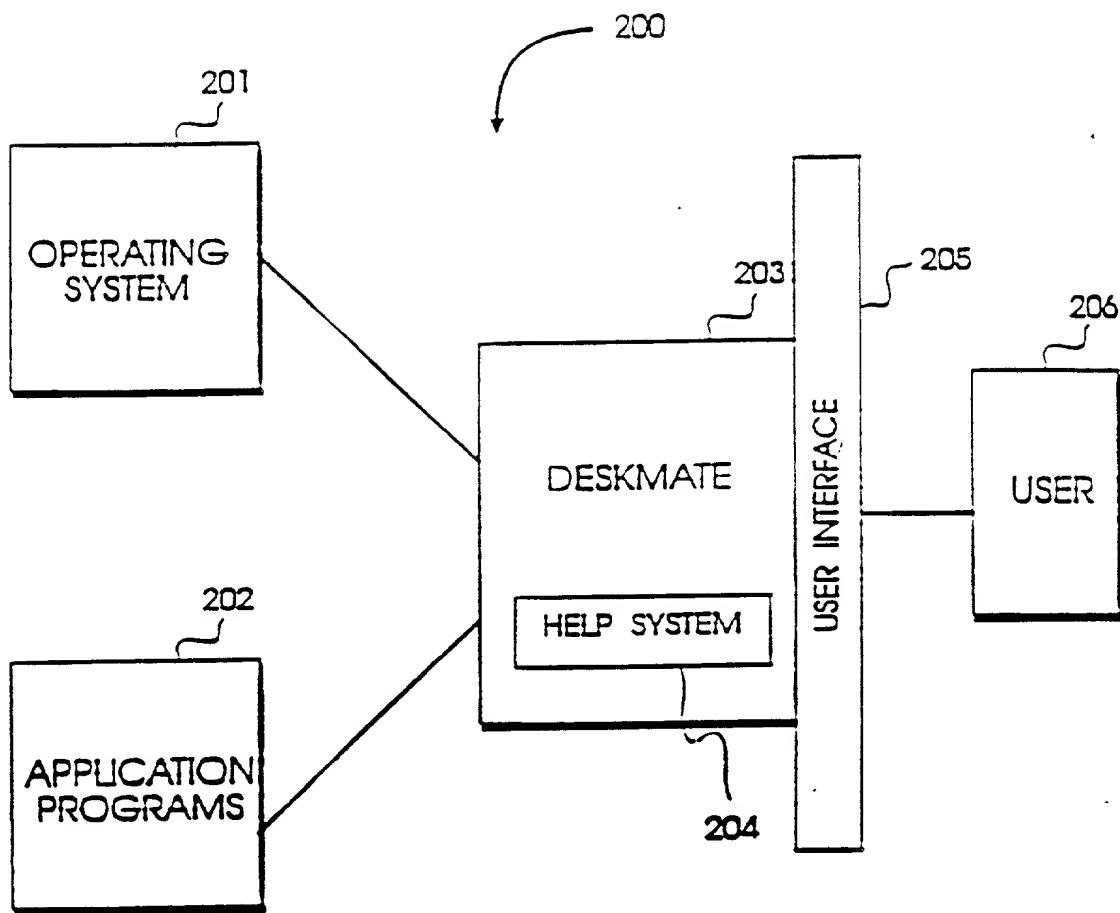


FIG. 2

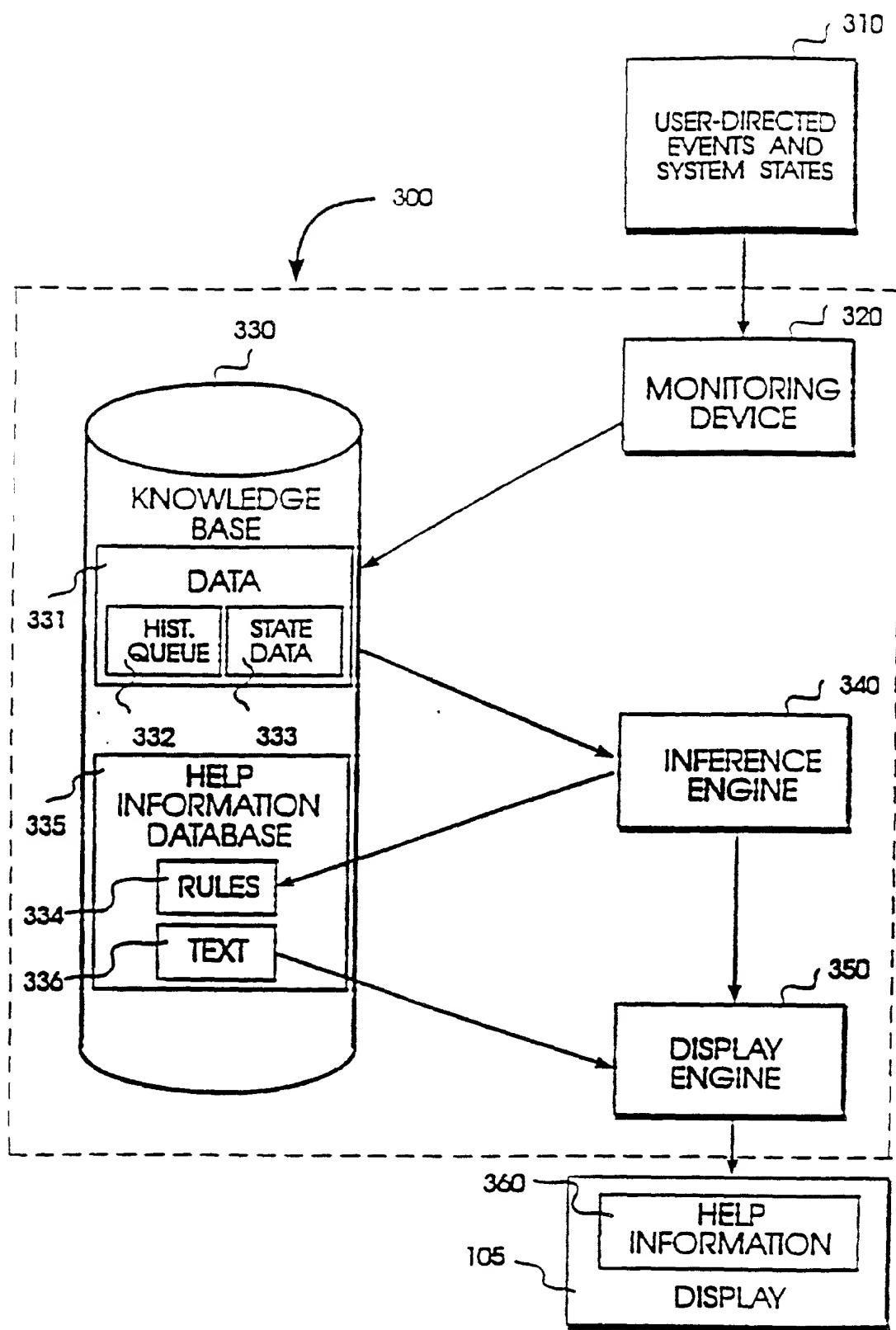


FIG. 3A

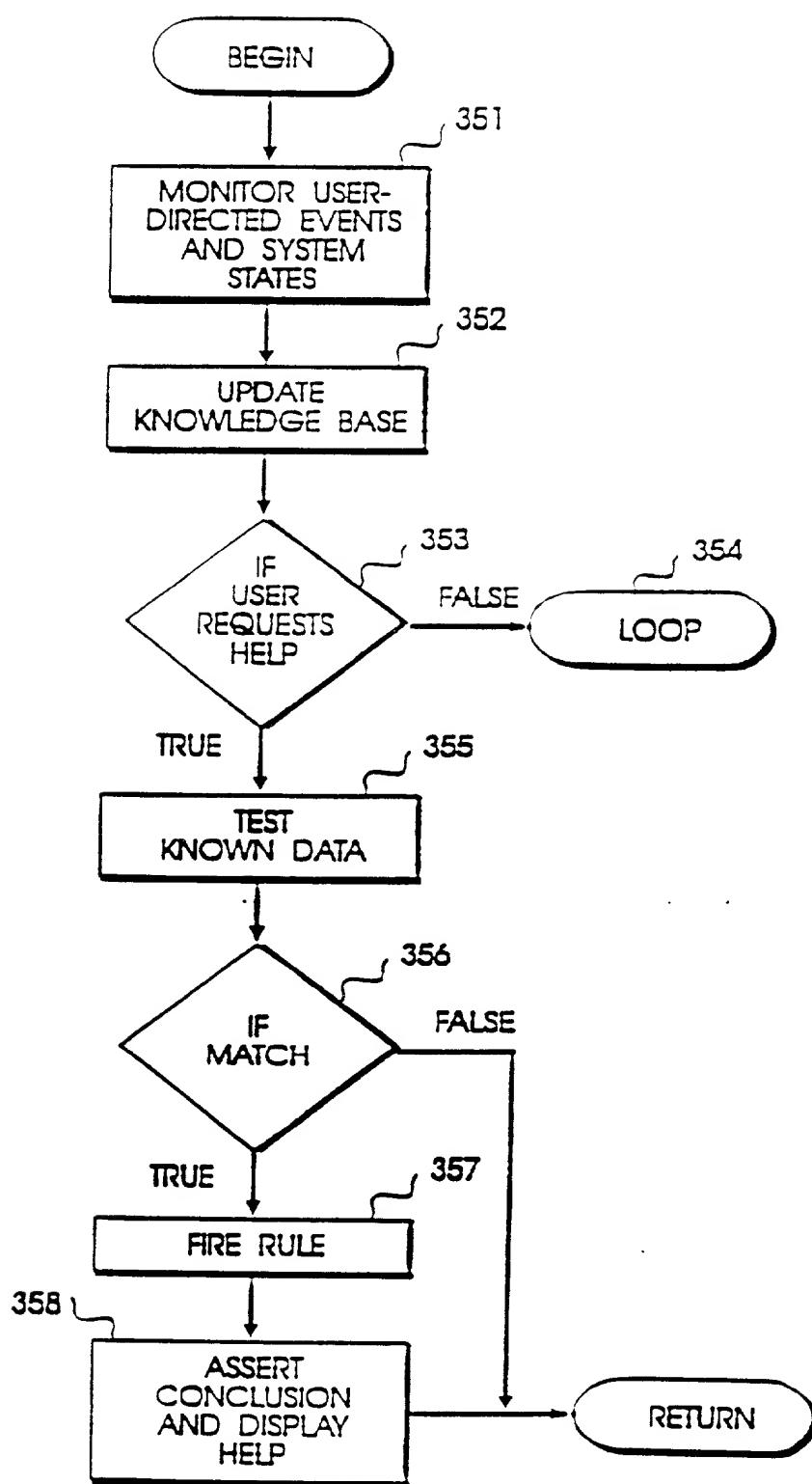


FIG. 3B

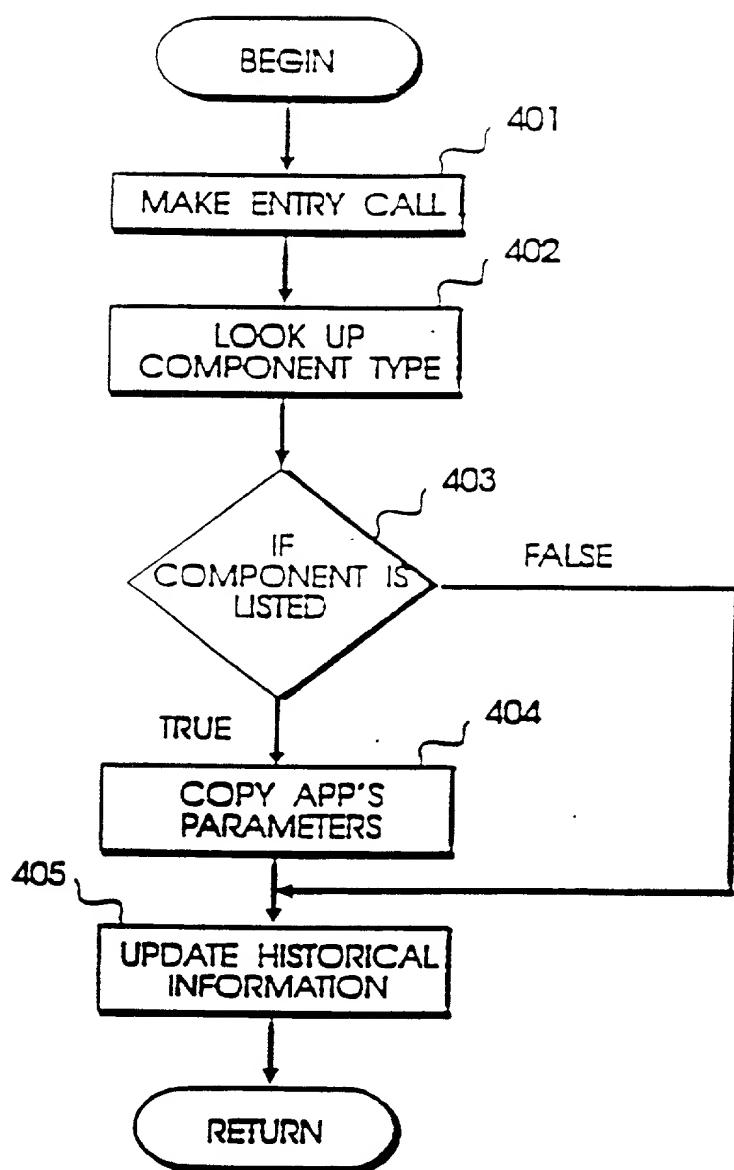


FIG. 4

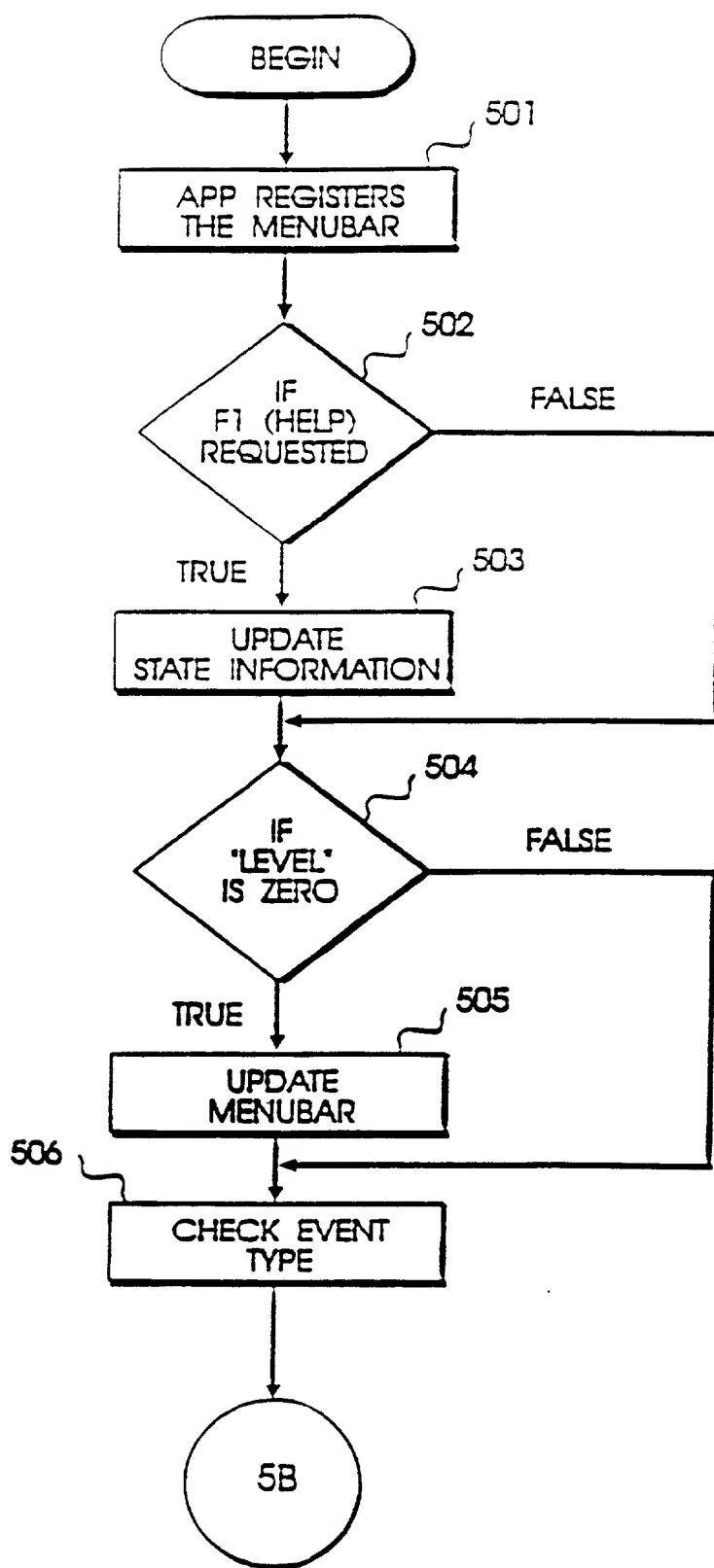


FIG. 5A

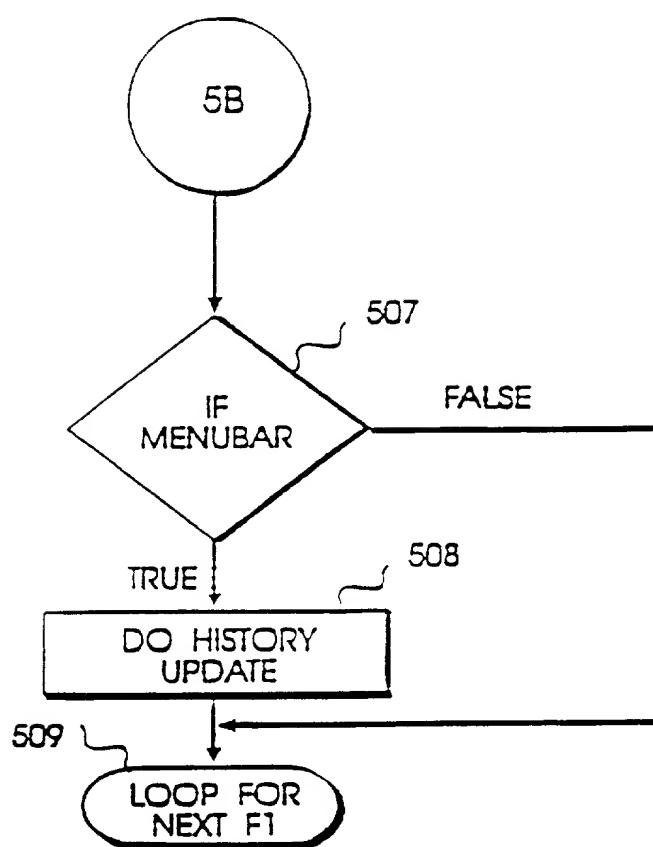


FIG. 5B

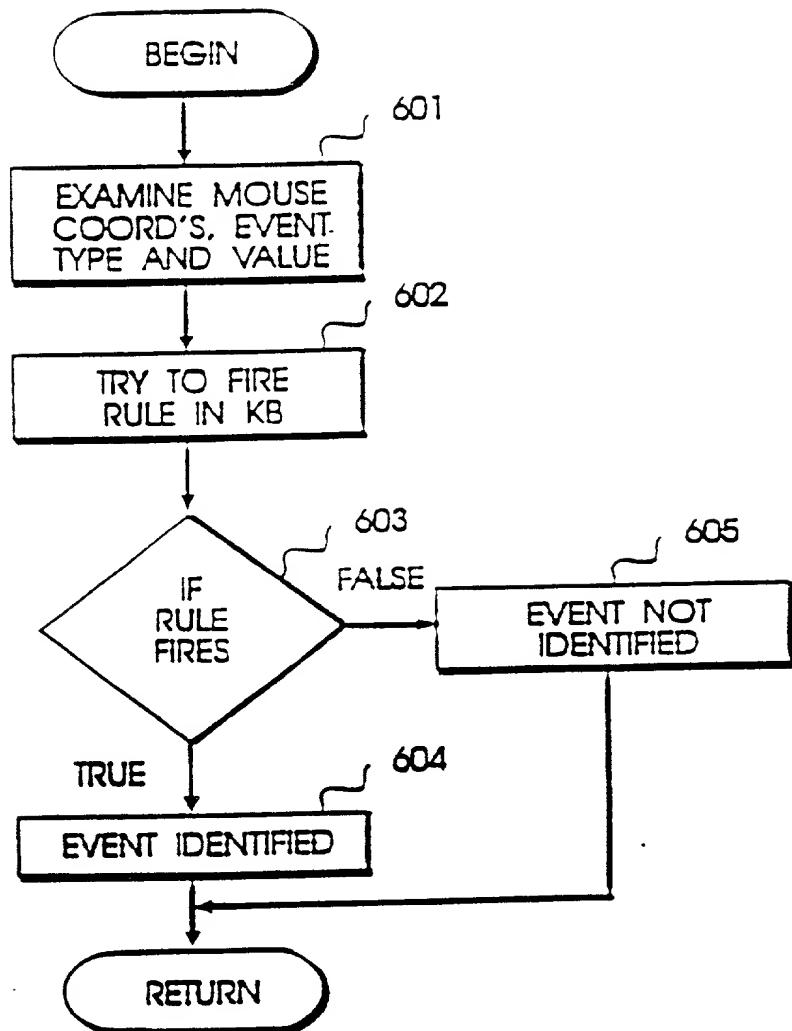


FIG. 6

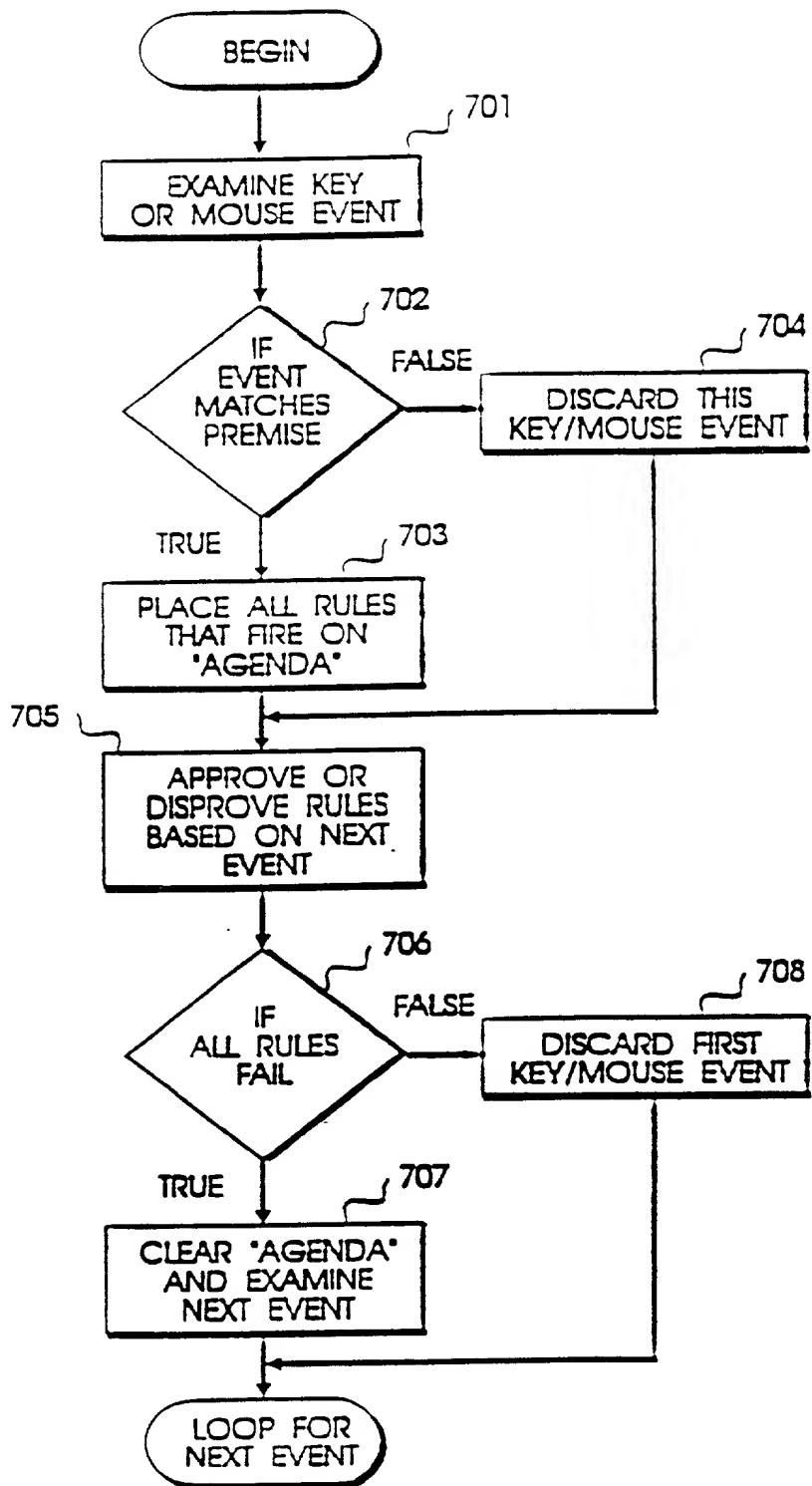


FIG. 7

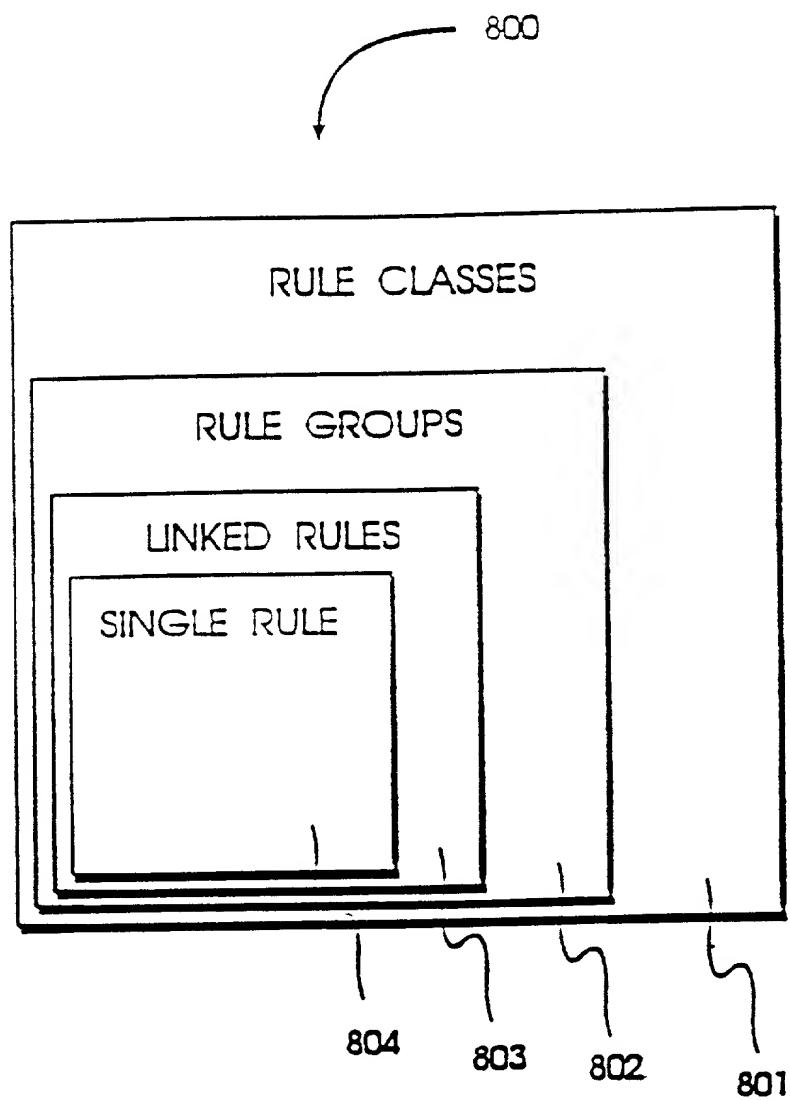


FIG. 8

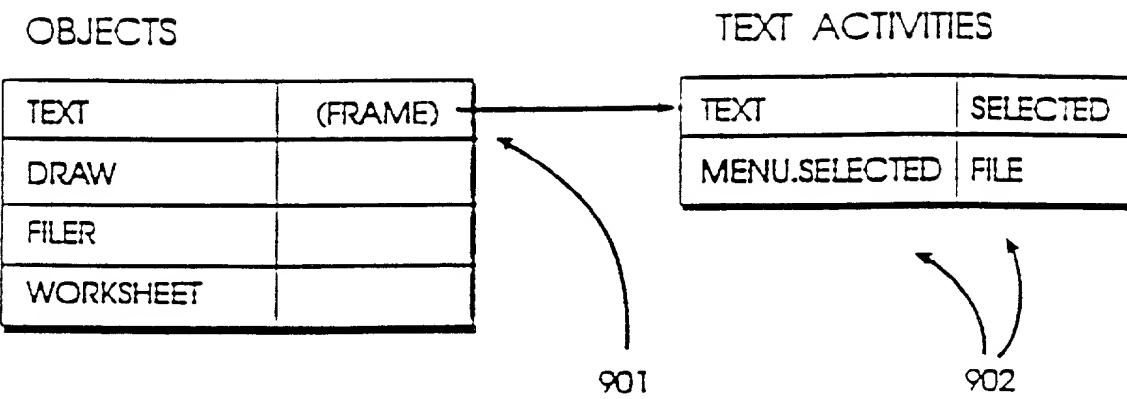


FIG. 9

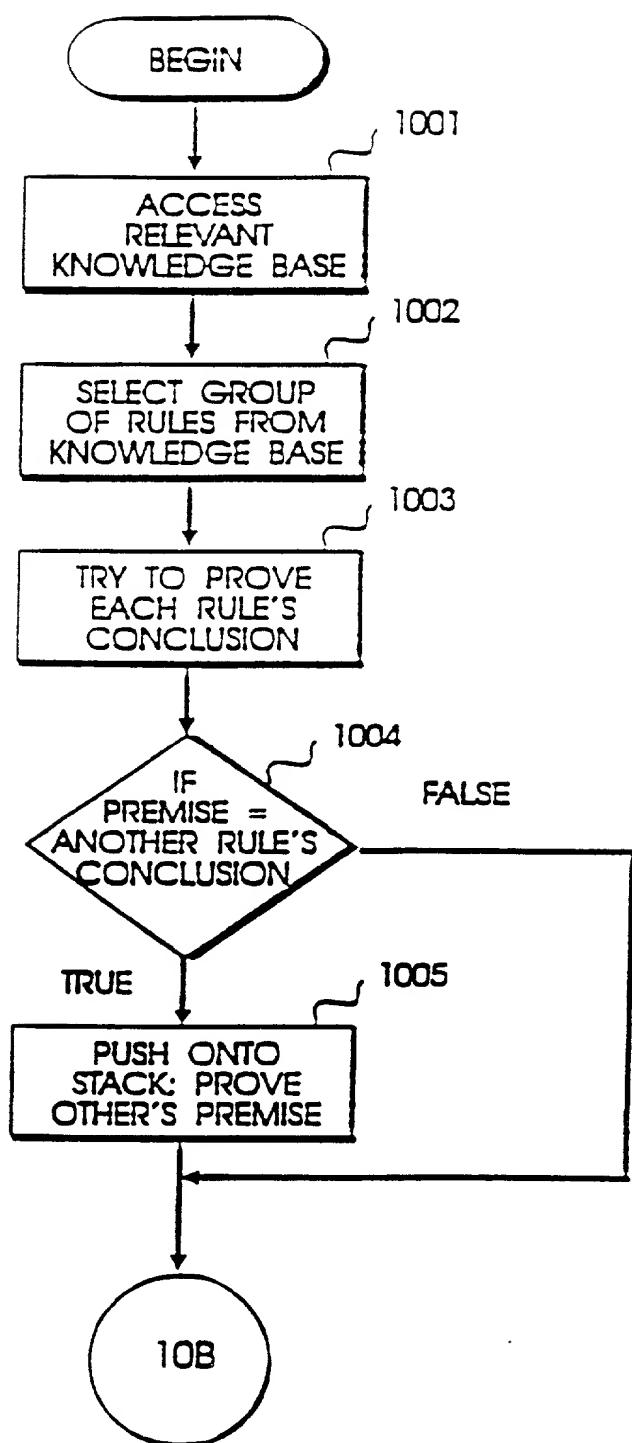


FIG. 10A

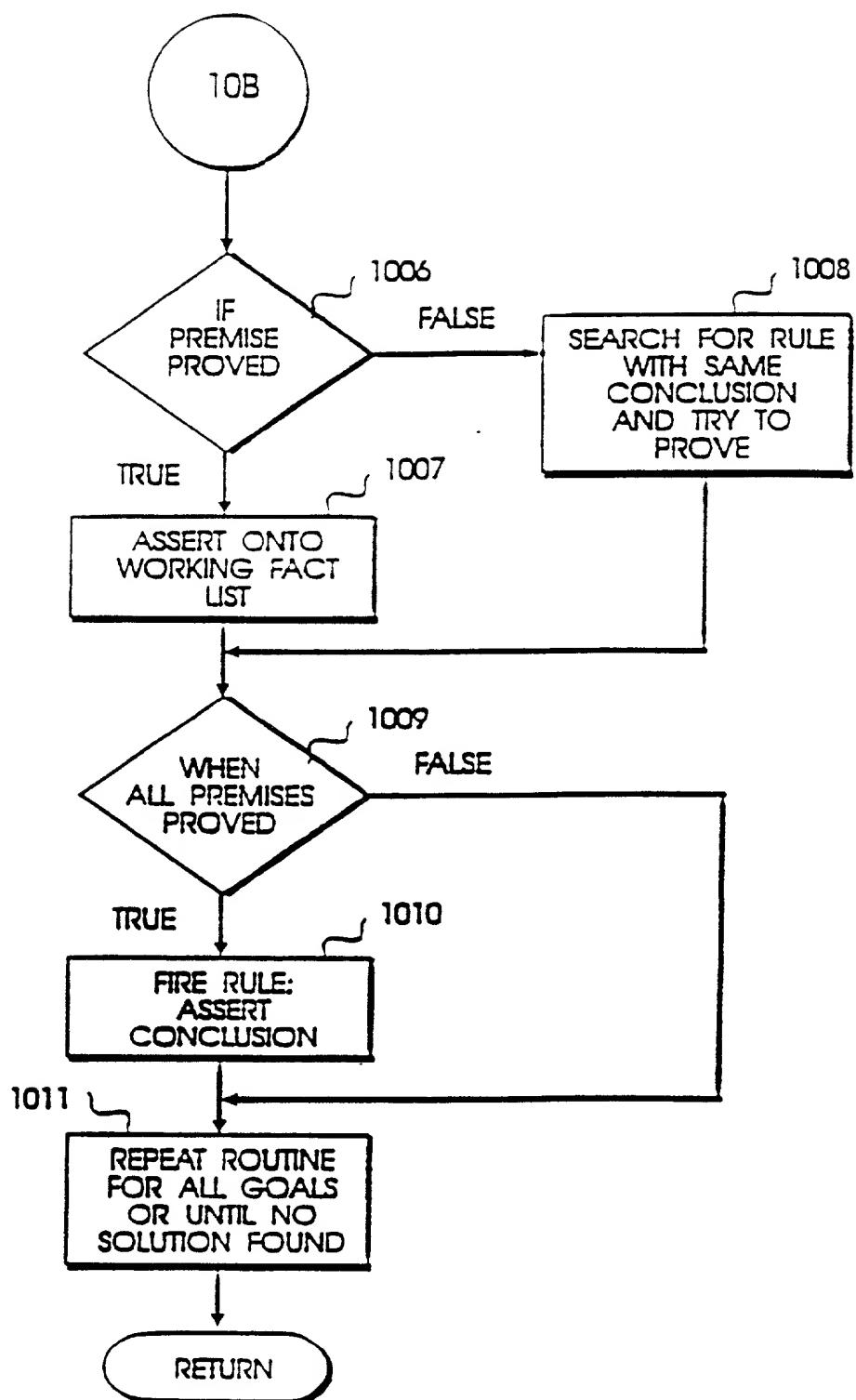


FIG. 10B

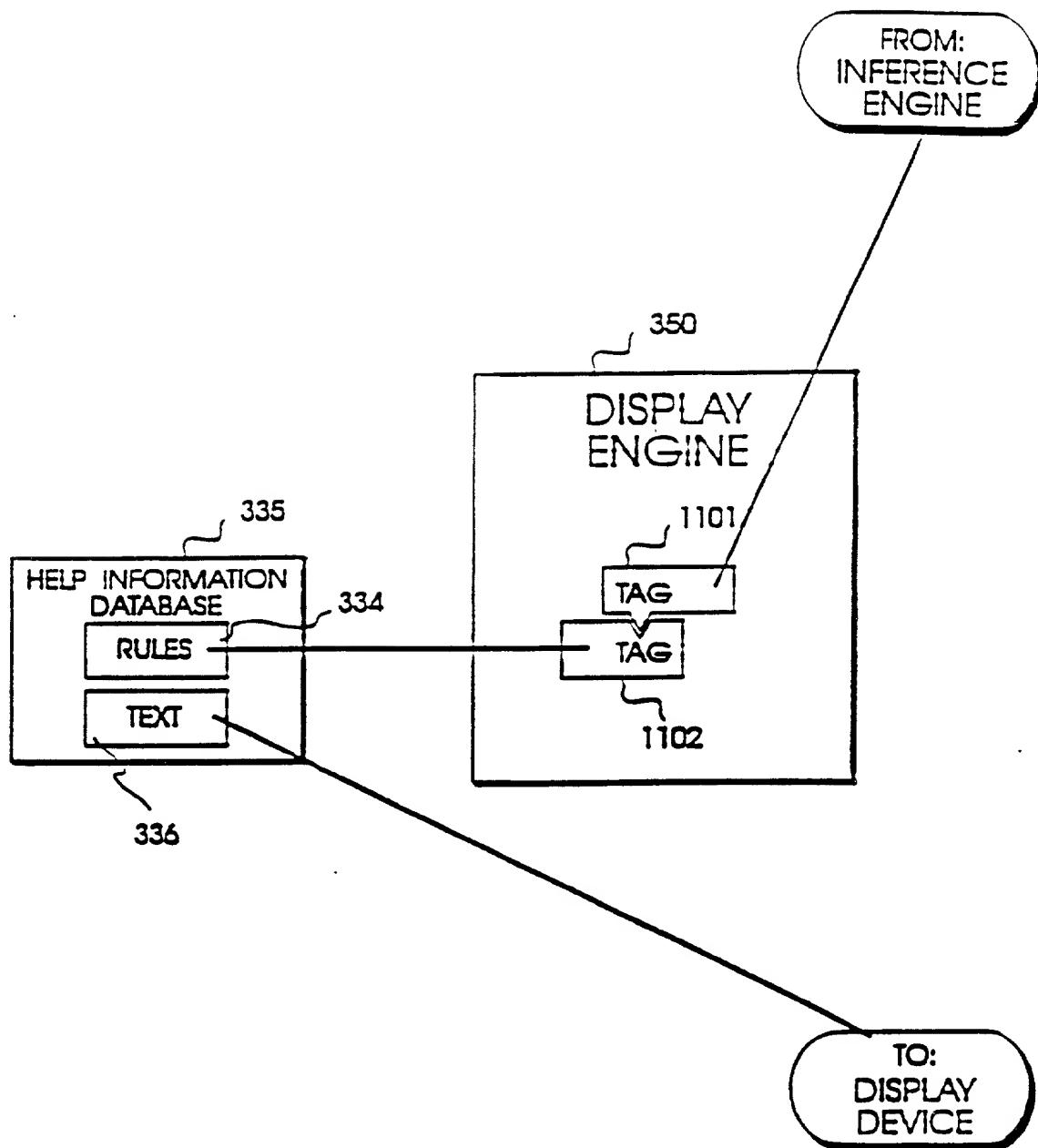


FIG. 11

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to:
Commissioner of Patents and Trademarks,
Washington, D.C. 20231, on 10-6-94

PATENT

By 

Attorney Docket No. 16265-11-2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:)
Charles D. Lanier, et al.)
Application for Reissue of) REISSUE DECLARATION UNDER 37
U.S. Patent 5,103,498) C.F.R. § 1.175(a)
granted April 7, 1992)
Filed: April 6, 1994)
For: INTELLIGENT HELP SYSTEM)

Commissioner of Patents and Trademarks
Washington, D.C. 20231

Sir:

We, Charles D. Lanier, Richard J. Wolf, and Leticia Villegas, declare as follows:

1. We are citizens of the United States of America and residents of 3035 Old Mill Run, Grapevine, Texas 76057; 1373 Autry Lane, Crowley, Texas 76036; and 8809 South Pointe Parkway East, Apt 2022, Phoenix AZ 85044, respectively.

2. The entire right, title, and interest to U.S. Patent No. 5,103,498, issued April 7, 1992, is vested in AST Research Inc. a Delaware corporation with a regular and established place of business in Irvine, California by virtue of an assignment from Tandy, Inc., recorded at Reel 6847, Frame 0109, Page 0002. Tandy, Inc., held right, title, and interest to U.S. Patent No. 5,103,498 by virtue of an assignment from us recorded at Reel 5440, Frame 0563.

3. We are the original, first and joint inventors of the invention described and claimed in the above-identified United States Letters Patent and the claims added by the above

referenced reissue application, for which invention we seek a reissue of the aforesaid Letters Patent.

4. We do not know and do not believe that said invention was ever known or used in the United States of America before our invention thereof.

5. We verily believe the original patent to be partly or wholly inoperative or invalid because of error without deceptive intent on our part. We believe the original patent to be partly or wholly inoperative or invalid because we claimed less than that to which we had a right to claim in the patent. In particular we believe that it was error not to include claims in the original patent such as claims 26 through 59 that are now added as reissue claims. The errors which resulted in the solicitation for reissue and resulted in such claims not being included in the original patent arose due to our failure to appreciate the extent to which the original claims included steps or elements which unduly limited the scope of protection afforded our invention. More specifically, in the original application, we failed to recognize and appreciate features and combinations of the invention which could stand on their own and were patentable over the prior art. The specific errors made in the original patent application and how and when they arose and under what circumstances they were discovered are discussed in the following paragraphs.

6. The error in claiming less than we had a right to claim in the original patent involves our failure to include a claim such as claim 26 now presented in this application for reissue. Claim 26 is a combination method claim that includes a number of method steps from claim 1 but that deletes from claim 1 unnecessary steps that unduly limit the protection afforded our invention by the original patent. Specifically, claim 26 eliminates the steps from claim 1 of "storing a plurality of rules...;", "detecting a request for help information from the

user;", "testing said rules...;", "selecting in response to the proved rules...;", and "displaying said selected help information...." Claim 26 adds the step "using the data indicating said series of user directed events stored in the knowledge base to select help information from said help information data base." The error of not including such a claim in the original application arose because of our failure to appreciate that the combination as recited in claim 26 described essential features of our invention without unnecessary limitations and was patentably distinct over the prior art. The circumstances under which we became aware of this error involved review of the original application by ourselves and our attorneys in light of allegations made in an interference declared between our original patent and a patent application filed by Nonaka, U.S. Patent Office Interference No. 103,022.

7. Claim 26 is further distinguishable from claim 1 in that claim 26 eliminates the language "said method operating independent of said computer program" that is present in claim 1. We believe that it was error not to include such a method claim without this language. We realized at the time we made the invention that a help system incorporating the steps recited in claim 1 could be implemented in such a way that it was either dependent or independent in operation from the computer program. In the original specification at column 12, beginning at line 28 we stated that "one skilled in the art could implement such a help system in another interface environment or without any interface environment." We did include in the original patent system claims for our invention, claim 15 and claim 24, which did not include the aforementioned language. We therefore believe that we had a right to claim a method such as claim 1 that did not include the aforementioned language and that our failure to do so was error. This error resulted from inadvertence without deceptive intent on our part. We have corrected this error by including method claim 26 without the aforementioned language, as well as a number of other claims discussed below in this reissue.

application. The circumstances under which we discovered this error involved review of the original application by ourselves and our attorneys in light of specific allegations made in an interference declared between our original patent and a patent application filed by Nonaka, U.S. Patent Office Interference No. 103,022. In the interference proceedings, party Nonaka made allegations that the aforementioned language limited our original method claims in a way that we did not intend or anticipate when we filed the original application. Upon our review of the original specification and the prior art, we believe that we have a right to method claims that do not include the aforementioned language.

8. We believe that the combination invention as recited in claim 26 is patentable over the prior art cited during prosecution of the original patent. Specifically, claim 26 includes the steps of "monitoring a series of user directed events from an input device;" and "generating data indicating said series of user directed events;" in combination with the other recited steps. The importance of these elements in overcoming the prior art was discussed in the amendment to the original patent application that we filed on June 26, 1991, beginning at page 14, second full paragraph and continuing through page 17, second full paragraph. We hereby incorporate into this declaration and adopt the above cited passage from the aforementioned amendment and believe the arguments made in those passages to distinguish claim 1 over the cited prior art are equally valid to distinguish claim 26 and all the new reissue claims that depend thereon over the cited prior art.

9. Claim 27 is identical to claim 2, but is written to depend on claim 26. We believe that it was error not to include such a claim in the original application for at least the reasons enumerated in paragraphs 6 and 7. This error arose due to the same reasons enumerated in paragraphs 5, 6, and 7. We further believe that claim 27 is patentable over the prior art

for at least the reasons we discussed in the amendment to the original patent application that we filed on June 26, 1991, beginning at page 17, third full paragraph and continuing through page 18, first full paragraph. We hereby incorporate into this declaration and adopt the above cited passage from the aforementioned amendment and believe the arguments made in those passages to distinguish claim 2 over the cited prior art are equally valid to distinguish claim 27 over the cited prior art. This error arose for at least the reasons enumerated in paragraphs 5, 6 and 7. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

10. Claim 28 is identical to claim 3, but depends on claim 27. We believe that it was error not to include such a claim 28 in the original application for at least the reasons enumerated in paragraphs 6 and 7. This error arose for at least the reasons stated in paragraphs 5, 6, and 7. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

11. Claim 29 includes the elements of claim 4 of the original patent with the exception of "checking if the user has requested help;" and is written to depend on claim 26. We believe not including such a claim in the original patent was error for at least the reasons enumerated in paragraph 6 and paragraph 7. This error arose for at least the reasons enumerated in paragraphs 5, 6 and 7. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

12. Claim 30 is identical to claim 8, but depends on claim 26. We believe that it was error not to include such a claim 30 in the original application for at least the reasons enumerated in paragraph 6 and 7. This error arose for at least the reasons enumerated in paragraphs 5, 6 and 7. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

13. Claim 31 depends on claim 26 and in addition recites a limitation where the using step uses "data indicating a series of user-directed events comprising at least two user-directed events." We believe it was error not to include such a claim 31 in the original patent for at least the reasons enumerated in paragraph 6 and paragraph 7. We believe this claim is patentable over the prior art for at least the reasons stated in paragraph 8. This error arose for at least the reasons enumerated in paragraphs 5, 6 and 7. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

14. Claim 32 depends on claim 26 and in addition recites a limitation where the using step uses "data indicating a series of user-directed events that may comprise four or more user-directed events." We believe it was error not to include such a claim in the original patent for at least the reasons enumerated in paragraph 6 and paragraph 7. We believe this claim is patentable over the prior art for at least the reasons stated in paragraph 8. This error arose for at least the reasons enumerated in paragraphs 5, 6 and 7. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

15. Claim 33 depends on claim 26 and in addition recites a limitation wherein the "series of user-directed events are events that are not necessarily related as being part of a particular command hierarchy." We believe it was error not to include such a claim in the original patent for at least the reasons enumerated in paragraph 6 and paragraph 7. We believe this claim is patentable over the prior art for at least the reasons stated in paragraph 8. This error arose for at least the reasons enumerated in paragraphs 5, 6 and 7. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

16. Claim 34 depends on claim 26 and in addition recites a step and limitation: "storing a plurality of rules for analyzing said generated data to determine appropriate help

information; and wherein said using step further comprises using an inference engine to test said rules against the data stored in the knowledge base to select appropriate help information." We believe it was error not to include such a claim in the original patent for at least the reasons enumerated in paragraph 6 and paragraph 7. This error arose for at least the reasons stated in paragraph 5. We believe this claim is patentable over the prior art for at least the reasons stated in paragraph 8. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

17. The error in claiming less than we had a right to claim in the original patent further involves failure to include a claim like or similar to claim 35 presented in this application for reissue. Claim 35 is a combination apparatus claim that includes a number of the elements from claim 15 but that deletes from claim 15 unnecessary elements and limitations that unduly limit the protection afforded our invention by the original patent. Specifically, claim 35 makes the changes from claim 15 described in the following paragraph.

*
18. In claim 35, the second element of claim 15 has been changed from "a display device" to "an output device." In addition, the seventh element of claim 15, "display engine means", has been deleted from claim 35 and the element "inference engine means" has been changed in claim 35 to include the functionality of "selecting help information for output by said output device to the user". We believe it was error not to include a claim such as claim 35 in the original patent because while computer systems generally will have a display device and display engine means, computer system configurations are possible which use neither a display device nor a display engine means such as computers using an audio output. These elements, which are common to a large number of computer systems, are not necessary to practice our invention nor are they necessary to overcome the prior art. Therefore we believe it was error not to

include a claim like claim 35 that recited a combination not including these unnecessary limitations. This error resulted from inadvertence on our part without deceptive intent. We believe this claim is patentable over the prior art for at least the reasons stated in paragraph 8 and we hereby incorporate into this declaration and adopt paragraph 3, page 18 from the amendment to the original patent application that we filed on June 26, 1991. We discovered this error under the circumstances discussed in paragraph 6.

19. We believe it was error not to include in the original patent a claim now presented as claim 36. Claim 36 includes all the elements of claim 15, but recites one additional element: a "selecting means" coupled to the help information data base that selects appropriate help information and then passes that information to the display engine. Furthermore, the display engine is recited to eliminate the functionality of selecting help information. We believe it was error not to include such a claim because while the specific embodiment of our invention described in the original patent includes a display engine with the functionality of selecting information from a help information data base, it would be obvious to anyone of skill in the software art that such functionality could just as easily be incorporated into a separate selecting means, leaving the display engine means with just the functionality of displaying information to the user. We believe that the recitation of the selecting function as being within the display engine means may unduly limit the scope of protection afforded our invention by claim 15. We therefore believe that it was error not to include such a claim as claim 36 in the original patent. This error resulted from our inadvertent failure to fully comprehend that the invention could be practiced without the limitations recited in the display engine means. We believe this claim is patentable over the prior art for at least the reasons stated in paragraph 8 and we hereby incorporate into this declaration and adopt paragraph 3, page 18 from the amendment to the original patent

application that we filed on June 26, 1991. We discovered this error under the circumstances discussed in paragraph 6.

20. Claim 37 is identical to claim 16, but is written to depend on new claims 35 and 36. We believe that it was error not to include a claim such as 37 for at least the reasons enumerated in paragraphs 18 and 19. This error arose for at least the reasons stated in paragraphs 5, 18, and 19. We discovered this error under the circumstances discussed in paragraph 6.

21. Claim 38 is identical to claim 17, but is written to depend on new claim 37. We believe that it was error not to include a claim such as 38 for at least the reasons enumerated in paragraphs 18 and 19. This error arose for at least the reasons stated in paragraphs 5, 18, and 19. We discovered this error under the circumstances discussed in paragraph 6.

22. Claim 39 is identical to claim 18, but is written to depend on new claims 35 and 36. We believe that it was error not to include a claim such as 39 for at least the reasons enumerated in paragraphs 18 and 19. This error arose for at least the reasons stated in paragraphs 5, 18, and 19. We discovered this error under the circumstances discussed in paragraph 6.

23. Claim 40 is identical to claim 19, but is written to depend on new claims 35 and 36. We believe that it was error not to include a claim such as 40 for at least the reasons enumerated in paragraphs 18 and 19. This error arose for at least the reasons stated in paragraphs 5, 18, and 19. We discovered this error under the circumstances discussed in paragraph 6.

24. Claim 41 is identical to claim 20, but is written to depend on new claims 35 and 36. We believe that it was error

not to include a claim such as 41 for at least the reasons enumerated in paragraphs 18 and 19. This error arose for at least the reasons stated in paragraphs 5, 18, and 19. We discovered this error under the circumstances discussed in paragraph 6.

25. Claim 42 is identical to claim 21, but is written to depend on new claims 35 and 36. We believe that it was error not to include a claim such as 42 for at least the reasons enumerated in paragraphs 18 and 19. This error arose for at least the reasons stated in paragraphs 5, 18, and 19. We discovered this error under the circumstances discussed in paragraph 6.

26. Claim 43 is identical to claim 22, but is written to depend on new claims 35 and 36. We believe that it was error not to include a claim such as 43 for at least the reasons enumerated in paragraphs 18 and 19. This error arose for at least the reasons stated in paragraphs 5, 18, and 19. We discovered this error under the circumstances discussed in paragraph 6.

27. Claim 44 is identical to claim 23, but is written to depend on new claims 35 and 36. We believe that it was error not to include a claim such as 44 for at least the reasons enumerated in paragraphs 18 and 19. This error arose for at least the reasons stated in paragraphs 5, 18, and 19. We discovered this error under the circumstances discussed in paragraph 6.

28. Claim 45 is identical to claim 24, but is written to depend on new claims 35 and 36. We believe that it was error not to include a claim such as 45 for at least the reasons enumerated in paragraphs 18 and 19. This error arose for at least the reasons stated in paragraphs 5, 18, and 19. We

discovered this error under the circumstances discussed in paragraph 6.

29. Claim 45 largely follows claim 1, but eliminates the language referred to in paragraph 7 and eliminates the step "detecting a request for help from the user". Although the step of detecting a request for help will be common to many types of help systems, detecting a request is not a necessary step in order to practice our invention and therefore represents an unnecessary step that may unduly limit the scope of protection afforded our invention. For example, the system could readily use the invention and provide help information to the user regardless of whether the user has specifically requested help information. We therefore believe that it was error not to include a claim such as claim 45 in the original patent application. We believe that this claim and all subsequent claims which depend thereon are patentable over the cited prior art for at least the reasons stated in paragraph 8. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

*
30. Claim 46 largely follows claim 1, but eliminates the step "detecting a request for help from the user." Although the step of detecting a request for help will be common to many types of help systems, detecting a request is not a necessary step in order to practice our invention and therefore represents an unnecessary step that may unduly limit the scope of protection afforded our invention. For example, the system could readily use the invention and provide help information to the user regardless of whether the user has specifically requested help information. We therefore believe that it was error not to include a claim in the original patent application which is now presented as claim 46. We believe that this claim, and all subsequent claims which depend thereon, to be patentable over the cited prior art for at least the reasons stated in paragraph 8.

We discovered this error under the circumstances discussed in paragraphs 6 and 7.

31. Claim 47 is identical to claim 2, but is rewritten to depend on new claims 45 and 46. We believe that it was error not to include a claim corresponding to claim 47 in the original patent application for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

32. Claim 48 is identical to claim 3, but is rewritten to depend on new claim 47. We believe that it was error not to include a claim corresponding to claim 48 in the original patent application for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

33. Claim 49 includes the elements of claim 4 of the original patent with the exception of "checking if the user has requested help;" and is written to depend on claims 45 and 46. We believe not including such a claim in the original patent was error for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5, 29, and 30. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

34. Claim 50 is identical to claim 5, but is rewritten to depend on new claims 45 and 46. We believe that it was error not to include a claim corresponding to claim 50 in the original patent application for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

35. Claim 51 is identical to claim 6, but is rewritten to depend on new claim 50. We believe that it was error not to include a claim corresponding to claim 51 in the original patent application for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

36. Claim 52 is identical to claim 7, but is rewritten to depend on new claim 50. We believe that it was error not to include a claim corresponding to claim 52 in the original patent application for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

37. Claim 53 is identical to claim 8, but is rewritten to depend on new claims 45 and 46. We believe that it was error not to include a claim corresponding to claim 53 in the original patent application for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

38. Claim 54 is identical to claim 9, but is rewritten to depend on new claims 45 and 46. We believe that it was error not to include a claim corresponding to claim 54 in the original patent application for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

39. Claim 55 is identical to claim 10, but is rewritten to depend on new claims 45 and 46. We believe that it was error not to include a claim corresponding to claim 55 in the original patent application for at least the reasons enumerated

in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

40. Claim 56 is identical to claim 11, but is rewritten to depend on new claims 45 and 46. We believe that it was error not to include a claim corresponding to claim 56 in the original patent application for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

41. Claim 57 is identical to claim 12, but is rewritten to depend on new claims 45 and 46. We believe that it was error not to include a claim corresponding to claim 57 in the original patent application for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

42. Claim 58 is identical to claim 13, but is rewritten to depend on new claims 45 and 46. We believe that it was error not to include a claim corresponding to claim 58 in the original patent application for at least the reasons enumerated in paragraph 29 and paragraph 30. This error arose for at least the reasons stated in paragraph 5. We discovered this error under the circumstances discussed in paragraphs 6 and 7.

43. Claim 59 is identical to claim 14 in the original patent except for deletion of the language discussed in paragraph 7. We believe that it was error not to include a claim corresponding to claim 59 in the original patent. We believe it was error not to include a claim such as 59 for at least the reasons we enumerated in paragraph 7. This error arose at least for the reasons we enumerated in paragraphs 5, 6, and 7. We

believe claim 59 to be patentable over the prior art for at least the reasons stated in paragraph 8.

44. We have reviewed and understand the contents of the attached specification and claims, including the new claims as presented in this application for reissue of the original Letters Patent.

45. We acknowledge the duty to disclose information of which we are aware and which is material to the examination of this application for reissue of the original Letters Patent in accordance with 37 C.F.R. § 1.56.

46. The aforementioned errors in claiming less than we had a right to claim arose without any deceptive intention on our part.

47. We hereby declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

48. This declaration may be executed in counterpart and by copy with the same force and effect as if executed as one complete and integrated document.

Charles D. Lanier, et al.
Reissue of U.S. Patent 5,103,498
Page 16

PATENT

Full name of first inventor: Charles D. Lanier

Inventor's signature: Charles D. Lanier

Date: 7/19/94 Country of Citizenship: U.S.A.

Residence: 3035 Old Mill Run Grapevine, TX 76057

Post Office Address: Same

Full name of second inventor: Richard J. Wolf

Inventor's signature: Richard J. Wolf

Date: _____ Country of Citizenship: U.S.A.

Residence: 1373 Autry Lane Crowley, TX 76036

Post Office Address: Same

Full name of third inventor: Leticia Villegas
Inventor's signature: Leticia Villegas
Date: 6/11/94 Country of Citizenship: U.S.A.
Residence: 8809 South Pointe Parkway East, Apt 2022, Phoenix AZ
85044
Post Office Address: Same

H:\home\lsj\16265\16265112.DEC